AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions and listings of claims in the application:

1. (Currently Amended) A method of processing data, using an object oriented programming language, comprising:

defining a class which supports an option data structure having, in instances of the class, references to option values without ~~preallocation~~ allocation of memory space for the full option values when the instance is created, the option data structure including a type description of the option values; and

compiling an operation on an option value in an instance of the class using the type description in the option data structure.

2. (Original) A method as claimed in claim 1 wherein the option data structure identifies change handler code that is executed when an option value changes.

3. (Original) A method as claimed in claim 2 wherein change handler code for one option is defined in different classes within a class inheritance hierarchy and the change handler code from each class is executed when the option value changes.

4. (Original) A method as claimed in claim 1 wherein the option data structure includes a default value, the method further comprising, in a get operation to an instance of the class, if an option value which applies to the instances has been set, getting the set option value and, if no value which applies has been set, getting the default value for the class.

5. (Currently Amended) A method as claimed in claim 1 comprising:

defining a first class with a first option data structure of a first form which supports, in instances of the first class, references to option values without ~~preallocation~~ allocation of memory space for the full option values when the instance is created;

defining a second class with a second option data structure of a second form which supports, in instances of the second class, references to option values without ~~preallocation~~ allocation of memory space for the full option values when the instance is created, the second form being different from the first form; and

during compilation, encoding an option operation as a method call to an object of the first class and to an object of the second class without regard to the form of the option data structure supported by the first class.

6. (Original) A method as claimed in claim 1 further comprising:

notifying objects of a change in an option value through a change handler identified by an option binding, the option binding being located by first searching a mapping data structure for a previously computed mapping to the option binding and, if no mapping was previously computed, by then computing the mapping to the option binding and storing the mapping in the mapping data structure.

7. (Original) A method as claimed in claim 1 wherein the option data structure comprises a linked list of option items having option values.

8. (Original) A method as claimed in claim 1 wherein a nonlocal option value applies to other objects in a nonlocal option hierarchy.

9. (Original) A method as claimed in claim 8 wherein the nonlocal option hierarchy is a graphical hierarchy.

10. (Original) A method as claimed in claim 1 wherein the class which supports the option data structure includes defined fields to support values in preallocated memory space.

11. (Original) A method as claimed in claim 1 wherein the type description is used to check the declared type of a value to be set in a set operation.

12. (Original) A method as claimed in claim 1 wherein the type description is used to check the legality of an operation to be performed on a value obtained in a get operation.

13. (Currently Amended) A data processing system using an object oriented programming language comprising:

a class which supports an option data structure having, in instances of the class, references to option values without ~~preallocation~~ allocation of memory space for the full option values when the instance is created, the option data structure including a type description of the option values; and

a compiler which, when compiling an operation on an option value in an instance of the class, uses the type description of the option value in the option data structure.

14. (Original) A system as claimed in claim 13 wherein the option data structure identifies change handler code that is executed when an option value changes.

15. (Original) A system as claimed in claim 14 wherein change handler code for one option is defined in different classes within a class inheritance hierarchy and the change handler code from each class is executed when the option value changes.

16. (Previously Presented) A system as claimed in claim 13 wherein the option data structure includes a default value which is obtained when no option value has been set in an applicable instance object.

17. (Original) A system as claimed in claim 13 comprising plural classes having data structures of different forms, and a compiler which encodes an option operation as a method call to an instance object of one of the classes without regard to the form of the option data structure supported by the class.

18. (Original) A system as claimed in claim 13 further comprising change handlers which notify objects of a change in an option value and a mapping data structure which maps an option name and class to an option binding which identifies a change handler.

19. (Original) A system as claimed in claim 13 wherein the option data structure comprises a linked list of option items having option values.

20. (Original) A system as claimed in claim 13 wherein a nonlocal option value applies to other objects in a nonlocal option hierarchy.

21. (Original) A system as claimed in claim 20 wherein the nonlocal option hierarchy is a graphical hierarchy.

22. (Original) A system as claimed in claim 13 wherein the class which supports the option data structure includes defined fields to support values in preallocated memory space.

23. (Currently Amended) A system as claimed in claim 13 wherein the type description is used to check the declared type of a value to be set in a set operation.

24. (Original) A system as claimed in claim 13 wherein the type description is used to check the legality of an operation to be performed on a value obtained in a get operation.

25. (Currently Amended) A data processing system using an object oriented programming language comprising:

means for defining a class which supports an option data structure having, in instances of the class, references to option values without ~~preallocation~~ allocation of memory space for the full option values when the instance is created, the option data structure including a type of description of the option values; and

compiler means for compiling an operation on an option value in an instance of the class using the type description in the option data structure.

26. (Currently Amended) A computer program product comprising:

a computer usable medium for storing data; and

a set of computer program instructions using an object oriented programming language embodied on the computer usable medium, including instructions to

define a class which supports an option data structure having, in instances of the class, references to option values without ~~preallocation~~ allocation of memory space for the full option values when the instance is created, the option data structure including a type description of the option values; and

compile an operation on the option value using the type description of the option value in the option data structure.

27. (Original) A product as claimed in claim 26 wherein the option data structure comprises a linked list of option items having option values.

28. (Cancelled).

29. (New) The method of claim 1, wherein the allocation of memory space occurs when the option value is set.

30. (New) The method of claim 1, wherein defining the class also supports a data structure having, in instances of the class, references to field values for which memory space is allocated when the instance is created, the data structure including a type description of the field value; and

compiling an operation on a field value in an instance of the class using the type description in the data structure.